

In the claims:

1. A program parallelization device comprising:
a control/data flow analysis unit which analyzes
the control flow and the data flow of a sequential
processing program;

5 a fork point candidate determination unit which
determines the fork point candidates of the sequential
processing program by referring to the results of the
analysis of the control flow and the data flow by said
control/data flow analysis unit;

10 a parallel execution performance evaluation unit
which evaluates, with respect to an input data, a
parallel execution performance when the sequential
processing program has been parallelized by a test
combination of fork point candidates that were given ;

15 a best fork point candidate combination
determination unit which generates a test combination of
the fork point candidates that were determined by said
fork point candidate determination unit, provides the
test combination to said parallel execution performance
20 evaluation unit, and by taking the parallel execution
performance of the test fork point candidate combination
evaluated thereby as the reference, determines the best
fork point candidate combination; and

a parallelized program output unit which
25 generates and outputs a parallelized program by

inserting a fork command at each fork point candidate of the best combination determined by said best fork point candidate combination determination unit.

2. A program parallelization device comprising:
a control/data flow analysis unit which analyzes the control flow and the data flow of a sequential processing program;

5 a fork point candidate determination unit which determines the fork point candidates of the sequential processing program by referring to the results of the analysis of the control flow and the data flow by said control/data flow analysis unit;

10 a parallel execution performance evaluation unit which evaluates, with respect to an input data, a parallel execution performance when the sequential processing program has been parallelized by a test combination of fork point candidates that were given;

15 a best fork point candidate combination determination unit which generates a test combination only consisting of the combination of fork point candidates that can be simultaneously executed in the one-time fork model from the fork point candidates determined by said fork point candidate determination unit, provides the test combination to said parallel execution performance evaluation unit, and by taking the
20 parallel execution performance of the test fork point

candidate combination evaluated thereby as the reference,
25 determines the best fork point candidate combination;

a parallelized program output unit which
generates and outputs a parallelized program by
inserting a fork command at each fork point candidate of
the best combination determined by said best fork point
30 candidate combination determination unit.

3. The program parallelization device as set forth
in claim 1,

wherein said parallel execution performance
evaluation unit

5 generates a sequential execution trace when the
sequential processing program was sequentially executed
with the input data, divides the sequential execution
trace by taking all the terminal point candidates as
division points, analyzes thread element information for
10 each thread element, and simulates parallel execution by
units of thread element with respect to the test
combination of the fork point candidates that were given
to calculate the parallel execution performance.

4. The program parallelization device as set forth
in claim 1,

wherein said best fork point candidate
combination determination unit

5 constructs a better combination by ranking the

fork point candidates determined by said fork candidate
determination unit in the order in which the fork point
candidates are predicted to have an influence on
parallel execution performance, and evaluating the
parallel execution performance according to this order
by taking the best fork point candidate combination at
that time as the reference.

5. The program parallelization device as set forth
in claim 4,

wherein said best fork point candidate
combination determination unit

assuming that the combination of the fork point
candidates including the prescribed numbers from the top
in the order of the fork point candidates determined is
an initial combination, evaluates the parallel execution
performance of the initial combination with said
parallel execution performance evaluation unit, and sets
the initial combination to the best fork point candidate
combination at this time.

6. The program parallelization device as set forth
in claim 1,

wherein said best fork point candidate
combination determination unit

divides said collection of all the fork point
candidates that have been determined by said fork point

candidate determination unit into fork point candidate groups in such a way that the fork point candidates have as little effects as possible on each other, generates a test fork point candidate combination for a group in the divided fork point candidate groups in which the best fork point candidate combination determination processing has not been performed, performs the best fork point candidate combination determination processing that determines the best fork point candidate combination by referring to the result of parallel execution performance of the test fork point candidate combination evaluated by said parallel execution performance evaluation unit, and determines the sum of the best fork point candidate combinations, which are the processing results for each group, as the overall processing result.

7. The program parallelization device as set forth in claim 6,

wherein said best fork point candidate combination determination unit

calls the fork point candidate group partition processing taking the collection of all the fork point candidates determined by said fork point candidate determination unit as a collection after the processing of said fork point candidate determination unit has been completed, when the fork point candidate group partition

processing is called, starts the group partition
processing of the collection if the number of fork point
candidates belonging to the given fork point candidate
collection is higher than the designated division number
15 lower limit, returns to the origin from where the fork
point candidate group partition processing was called
without performing the group partition processing if the
number of the fork point candidates is lower, divides
from the collection the fork point candidate collections
20 in which the number of fork point candidates that cancel
themselves is higher than the designated number to
generate a new group, further divides the collection
into two groups, recursively calls the fork point
candidate group partition processing taking one group as
25 a collection and performs group partitioning of the
group, recursively calls the fork point candidate group
partitioning process taking the other group as a
collection and performs group partitioning of the group,
returns to the origin from where the fork point
30 candidate group partitioning process was called,
performs the best fork point candidate combination
determination processing for the groups in which the
best fork point candidate combination determination
processing has not been performed among the groups of
35 fork point candidates that were divided, determines
whether the processing of all the groups has been
completed, if there is a group that has not been

processed, reiterates the best fork point candidate
combination determination processing for the groups that
40 have not been processed, and, when the processing of all
the groups has been completed, outputs the sum of the
fork point candidate combination, which is the result of
processing for each group, as the overall result.

8. A program parallelization method for a
multithreading method in which a sequential processing
program is divided into a plurality of threads and a
plurality of processors execute the threads in parallel,
5 comprising the steps of:

analyzing the control flow and the data flow of a
sequential processing program;

determining the fork point candidates of the
sequential processing program by referring to the
10 results of the analysis of the control flow and the data
flow;

generating a test fork point candidate
combination from the determined fork point candidates;

evaluating, with respect to an input data, the
15 parallel execution performance when the sequential
processing program has been parallelized by the
generated test fork point candidate combination;

determining a best fork point candidate
combination by taking the parallel execution performance
20 of the evaluated test fork point candidate combination

as the reference;

inserting a fork command at each fork point candidate in the determined best combination to generate and output a parallelized program.

25

9. A program parallelization method for a multithreading method in which a sequential processing program is divided into a plurality of threads and a plurality of processors execute the threads in parallel, comprising the steps of:

5

analyzing the control flow and the data flow of a sequential processing program;

determining the fork point candidates of the sequential processing program by referring to the results of the analysis of the control flow and the data flow;

10

generating a test combination only consisting of the fork point candidates that can be simultaneously executed in the one-time fork model from the determined fork point candidates;

15

evaluating, with respect to an input data, the parallel execution performance when the sequential processing program has been parallelized by the generated test fork point candidate combination;

20

determining a best fork point candidate combination by taking the parallel execution performance of the evaluated test fork point candidate combination

as the reference;

25 inserting a fork command at each fork point
candidate in the determined best combination to generate
and output a parallelized program.

10. A program parallelization method for a
multithreading method in which a sequential processing
program is divided into a plurality of threads and a
plurality of processors execute the threads in parallel,
5 comprising:

 a step in which a control/data flow analysis unit
analyzes the control flow and the data flow of a
sequential processing program;

10 a step in which a fork point candidate
determination unit generates the fork point candidates
by referring to the results of the analysis of the
control flow and the data flow by the control/data flow
analysis unit;

15 a step in which a best fork point candidate
combination determination unit predicts the effect of
each of all the fork point candidates on the parallel
execution performance and ranks the fork point
candidates in the order of the effect;

20 a step in which the best fork point candidate
combination determination unit generates an initial fork
point candidate combination whose parallel execution
performance is evaluated first, and which is assumed to

be the best fork point candidate combination;

25 a step in which a parallel execution performance
evaluation unit generates a sequential execution trace
when the sequential processing program was sequentially
executed with the input data;

30 a step in which the parallel execution
performance evaluation unit divides the sequential
execution trace by taking all the terminal point
candidates as division points

35 a step in which the parallel execution
performance evaluation unit analyzes thread element
information for each thread element, and memorizes the
thread element information for each thread element.

 a step in which a best fork point candidate
combination determination unit selects one fork point
candidate that is ranked highest order among the non-
selected fork point candidates;

40 a step in which the best fork point candidate
combination determination unit assesses whether the fork
point candidate selected is contained in the best fork
point candidate combination;

45 a step in which, if the selected fork point
candidate is not contained in the best fork point
candidate combination, the best fork point candidate
combination determination unit adds the selected fork
point candidate to the best fork point candidate
combination, and sets the fork point candidate

50 combination as a test combination;

a step in which, if the selected fork point candidate is contained in the best fork point candidate combination, the best fork point candidate combination determination unit removes the selected fork point candidate from the best fork point candidate combination, and sets the fork point candidate combination as the test combination;

a step in which the best fork point candidate combination determination unit evaluates the parallel execution performance of parallelization by the test combination through the parallel execution performance evaluation unit;

a step in which the best fork point candidate combination determination unit compares the parallel execution performance of the test combination with the parallel execution performance of the best combination;

a step in which, if the parallel execution performance of the test combination is better, the best fork point candidate combination determination unit sets the test combination as the best fork point candidate combination at the current time;

a step in which the best fork point candidate combination determination unit assesses whether a fork point candidate that has not been selected exists, and if a fork point candidate that has not been selected exists, reiterates execution;

a step in which, if a non-selected fork point candidate does not exist, the best fork point candidate combination determination unit assesses whether a new
80 best fork point candidate combination is found in the previous iterative execution;

a step in which, if a new best fork point candidate combination is found, the best fork point candidate combination determination unit sets all the
85 fork point candidates to the non-selected state for the iterative execution;

a step in which, if a new best fork point candidate combination is not found, the best fork point candidate combination determination unit outputs the
90 determined best fork point candidate combination as the result of the best fork point candidate combination determination processing; and

a step in which a parallelized program output unit generates and outputs the parallelized program by
95 inserting a fork command at each fork point candidate of the best combination determined by the best fork point candidate combination determination unit.

11. A program parallelization method for a multithreading method in which a sequential processing program is divided into a plurality of threads and a plurality of processors execute the threads in parallel,
5 comprising:

a step in which a control/data flow analysis unit analyzes the control flow and the data flow of a sequential processing program;

10 a step in which a fork point candidate determination unit generates the fork point candidates by referring to the results of the analysis of the control flow and the data flow by the control/data flow analysis unit;

15 a step in which a best fork point candidate combination determination unit predicts the effect of each of all the fork point candidates on the parallel execution performance and ranks the fork point candidates in the order of the effect;

20 a step in which the best fork point candidate combination determination unit generates an initial fork point candidate combination whose parallel execution performance is evaluated first, and which is assumed to be the best fork point candidate combination;

25 a step in which a parallel execution performance evaluation unit generates a sequential execution trace when the sequential processing program was sequentially executed with the input data;

30 a step in which the parallel execution performance evaluation unit divides the sequential execution trace by taking all the terminal point candidates as division points;

a step in which the parallel execution

performance evaluation unit analyzes thread element
information for each thread element, and memorizes the
thread element information for each thread element;

a step in which a best fork point candidate
combination determination unit selects one fork point
candidate that is ranked highest order among the non-
selected fork point candidates;

a step in which the best fork point candidate
combination determination unit assesses whether the
selected fork point candidate is contained in the best
fork point candidate combination;

a step in which, if the selected fork point
candidate is not contained in the best fork point
candidate combination, the best fork point candidate
combination determination unit adds the selected fork
point candidate to the best fork point candidate
combination, and sets the fork point candidate
combination as a test combination;

a step in which the best fork point candidate
combination determination unit removes the fork point
candidate that cancels the selected fork point candidate
from the test combination;

a step in which, if the selected fork point
candidate is contained in the best fork point candidate
combination, the best fork point candidate combination
determination unit removes the selected fork point
candidate from the best fork point candidate combination,

60 and sets the fork point candidate combination as the
test combination;

 a step in which the best fork point candidate
combination determination unit evaluates the parallel
execution performance of parallelization by the test
65 combination through the parallel execution performance
evaluation unit;

 a step in which the best fork point candidate
combination determination unit compares the parallel
execution performance of the test combination with the
70 parallel execution performance of the best combination;

 a step in which, if the parallel execution
performance of the test combination is better, the best
fork point candidate combination determination unit
removes the fork point candidate that is canceled by the
75 selected fork point candidate from the test combination;

 a step in which the best fork point candidate
combination determination unit sets the test combination
as the best fork point candidate combination at the
current time;

80 a step in which the best fork point candidate
combination determination unit assesses whether a fork
point candidate that has not been selected exists, and
if a fork point candidate that has not been selected
exists, reiterates execution;

85 a step in which, if a non-selected fork point
candidate does not exist, the best fork point candidate

combination determination unit assesses whether a new best fork point candidate combination is found in the previous iterative execution;

90 a step in which, if a new best fork point candidate combination is found, the best fork point candidate combination determination unit sets all the fork point candidates to the non-selected state for the iterative execution;

95 a step in which, if a new best fork point candidate combination is not found, the best fork point candidate combination determination unit outputs the determined best fork point candidate combination to the parallelized program output unit as the result of the
100 best fork point candidate combination determination processing; and

 a step in which the parallelized program output unit generates and outputs the parallelized program by inserting a fork command at each fork point candidate of
105 the best combination determined by the best fork point candidate combination determination unit.

12. The program parallelization method as set forth in claim 8,

 wherein the step of evaluating said parallel execution performance

5 generates a sequential execution trace when the sequential processing program was sequentially executed

with the input data, divides the sequential execution trace by taking all the terminal point candidates as division points, analyzes thread element information for each thread element, and simulates parallel execution by units of thread element with respect to the test combination of the fork point candidates that were given to calculate the parallel execution performance.

13. The program parallelization method as set forth in claim 8,

wherein the step of determining the best combination of said fork point candidates

constructs a better combination by ranking the fork point candidates determined by said fork candidate determination unit in the order in which the fork point candidates are predicted to have an influence on parallel execution performance, and evaluating the parallel execution performance according to the order by taking the best fork point candidate combination at that time as the reference.

14. The program parallelization method as set forth in claim 8,

wherein the step of determining the best combination of said fork point candidates

divides the collection of all the fork point candidates into fork point candidate groups in such a

way that the fork point candidates have as little effects as possible on each other, generates a test fork point candidate combination for a group in the divided fork point candidate groups in which the best fork point candidate combination determination processing has not been performed, performs the best fork point candidate combination determination processing that determines the best fork point candidate combination by referring to the result of parallel execution performance of the test fork point candidate combination evaluated with respect to an input data, and determines the sum of the best fork point candidate combinations, which are the processing results for each group, as the overall processing result.

15. The program parallelization method as set forth in claim 8,

wherein the step of determining the best combination of said fork point candidates

calls the fork point candidate group partition processing taking the collection of all the fork point candidates determined by the fork point candidate determination unit as a collection after the processing of said fork point candidate determination unit has been completed, when the fork point candidate group partition processing is called, starts the group partition processing of the collection if the number of fork point

candidates belonging to the given fork point candidate collection is higher than the designated division number lower limit, returns to the origin from where the fork point candidate group partition processing was called without performing the group partition processing if the number of the fork point candidates is lower, divides from the collection the fork point candidate collections in which the number of fork point candidates that cancel themselves is higher than the designated number to generate a new group, further divides the collection into two groups, recursively calls the fork point candidate group partition processing taking one group as a collection and performs group partitioning of the group, recursively calls the fork point candidate group partitioning process taking the other group as a collection and performs group partitioning of the group, returns to the origin from where the fork point candidate group partitioning process was called, performs the best fork point candidate combination determination processing for the groups in which the best fork point candidate combination determination processing has not been performed among the groups of fork point candidates that were divided, determines whether the processing of all the groups has been completed, if there is a group that has not been processed, reiterates the best fork point candidate combination determination processing for the groups that

40 have not been processed, and, when the processing of all
the groups has been completed, outputs the sum of the
fork point candidate combination, which is the result of
processing for each group, as the overall result.

16. A program parallelization program that is
executed on a computer causing,

 a computer to operate as

 a control/data flow analysis function which
5 analyzes the control flow and the data flow of a
sequential processing program;

 a fork point candidate determination function
which determines the fork point candidates of the
sequential processing program by referring to the
10 results of the analysis of the control flow and the data
flow by said control/data flow analysis function;

 a parallel execution performance evaluation
function which evaluates, with respect to an input data,
a parallel execution performance when the sequential
15 processing program has been parallelized by a test
combination of fork point candidates that were given;

 a best fork point candidate combination
determination function which generates a test
combination of the fork point candidates that were
20 determined by said fork point candidate determination
function, provides the test combination to said parallel
execution performance evaluation function, and by taking

the parallel execution performance of the test fork
point candidate combination evaluated thereby as the
25 reference determines the best fork point candidate
combination; and

a parallelized program output function which
generates and outputs a parallelized program by
inserting a fork command at each fork point candidate of
30 the best combination determined by said best fork point
candidate combination determination function.

17. A program parallelization program that is
executed on a computer causing,

a computer to operate as

a control/data flow analysis function which
5 analyzes the control flow and the data flow of a
sequential processing program;

a fork point candidate determination function
which determines the fork point candidates of the
sequential processing program by referring to the
10 results of the analysis of the control flow and the data
flow by said control/data flow analysis function;

a parallel execution performance evaluation
function which evaluates, with respect to an input data,
a parallel execution performance when the sequential
15 processing program has been parallelized by a test
combination of fork point candidates that were given;

a best fork point candidate combination

determination function which generates a test
combination only consisting of the combination of fork
point candidates that can be simultaneously executed in
the one-time fork model from the fork point candidates
determined by said fork point candidate determination
function, provides the test combination to said parallel
execution performance evaluation function, and by taking
the parallel execution performance of the test fork
point candidate combination evaluated thereby as the
reference, determines the best fork point candidate
combination;

a parallelized program output function which
generates and outputs a parallelized program by
inserting a fork command at each fork point candidate of
the best combination determined by said best fork point
candidate combination determination function.

18. The program parallelization program as set forth
in claim 16,

wherein said parallel execution performance
evaluation function

generates a sequential execution trace when the
sequential processing program was sequentially executed
with the input data, divides the sequential execution
trace by taking all the terminal point candidates as
division points, analyzes thread element information for
each thread element, and simulates parallel execution by

units of thread element with respect to the test combination of the fork point candidates that were given to calculate the parallel execution performance.

19. The program parallelization program as set forth in claim 16,

wherein said best fork point candidate combination determination function

5 constructs a better combination by ranking the fork point candidates determined by said fork candidate determination function in the order in which the fork point candidates are predicted to have an influence on parallel execution performance, and evaluating the
10 parallel execution performance according to the order by taking the best fork point candidate combination at that time as the reference.

20. The program parallelization program as set forth in claim 19,

wherein said best fork point candidate combination determination function

5 assuming that the combination of the fork point candidates including the prescribed numbers from the top in the order of the fork point candidates determined is an initial combination, evaluates the parallel execution performance of the initial combination with said
10 parallel execution performance evaluation function, and

sets the initial combination to the best fork point candidate combination at this time.

21. The program parallelization program as set forth in claim 16,

wherein said best fork point candidate combination determination function

5 divides the collection of all the fork point candidates that have been determined by said fork point candidate determination function into fork point candidate groups in such a way that the fork point candidates have as little effects as possible on each other, generates a test fork point candidate combination for a group in the divided fork point candidate groups in which the best fork point candidate combination determination processing has not been performed, performs the best fork point candidate combination determination processing that determines the best fork point candidate combination by referring to the result of parallel execution performance of the test fork point candidate combination evaluated by said parallel execution performance evaluation function, and
10 determines the sum of the best fork point candidate combinations, which are the processing results for each group, as the overall processing result.

22. The program parallelization program as set forth

in claim 21,

wherein said best fork point candidate
combination determination function

5 calls the fork point candidate group partition
processing taking the collection of all the fork point
candidates determined by said fork point candidate
determination function as a collection after the
processing of said fork point candidate determination
10 function has been completed, when the fork point
candidate group partition processing is called, starts
the group partition processing of the collection if the
number of fork point candidates belonging to the given
fork point candidate collection is higher than the
15 designated division number lower limit, returns to the
origin from where the fork point candidate group
partition processing was called without performing the
group partition processing if the number of the fork
point candidates is lower, divides from the collection
20 the fork point candidate collections in which the number
of fork point candidates that cancel themselves is
higher than the designated number to generate a new
group, further divides the collection into two groups,
recursively calls the fork point candidate group
25 partition processing taking one group as a collection
and performs group partitioning of the group,
recursively calls the fork point candidate group
partitioning process taking the other group as a

collection and performs group partitioning of the group,
30 returns to the origin from where the fork point
candidate group partitioning process was called,
performs the best fork point candidate combination
determination processing for the groups in which the
best fork point candidate combination determination
35 processing has not been performed among the groups of
fork point candidates that were divided, determines
whether the processing of all the groups has been
completed, if there is a group that has not been
processed, reiterates the best fork point candidate
40 combination determination processing for the groups that
have not been processed, and, when the processing of all
the groups has been completed, outputs the sum of the
fork point candidate combination, which is the result of
processing for each group, as the overall result.

45